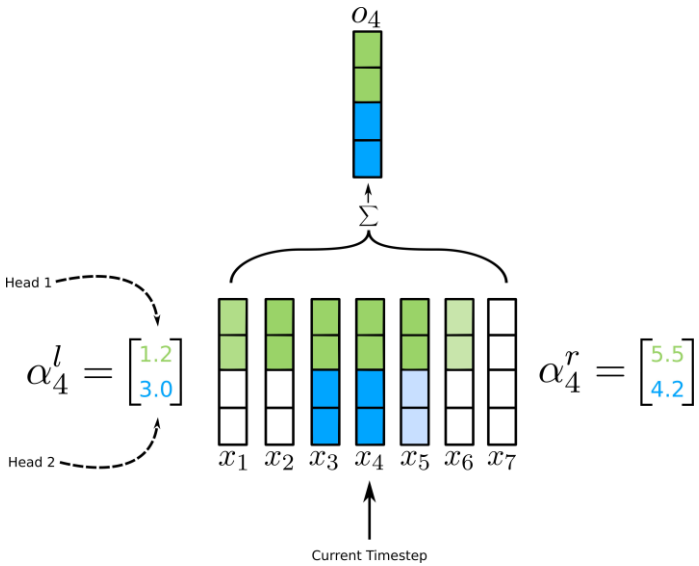
Time-aware Large Kernel Convolutions

Vasileios Lioutas and Yuhong Guo

Carleton UNIVERSITY

Brief Overview

- In this work, we introduce a novel sequence modeling approach called TaLK convolutions that is not based on self-attention.
- The proposed method has $O(n)$ time complexity and it uses an adaptive summation convolution kernel.

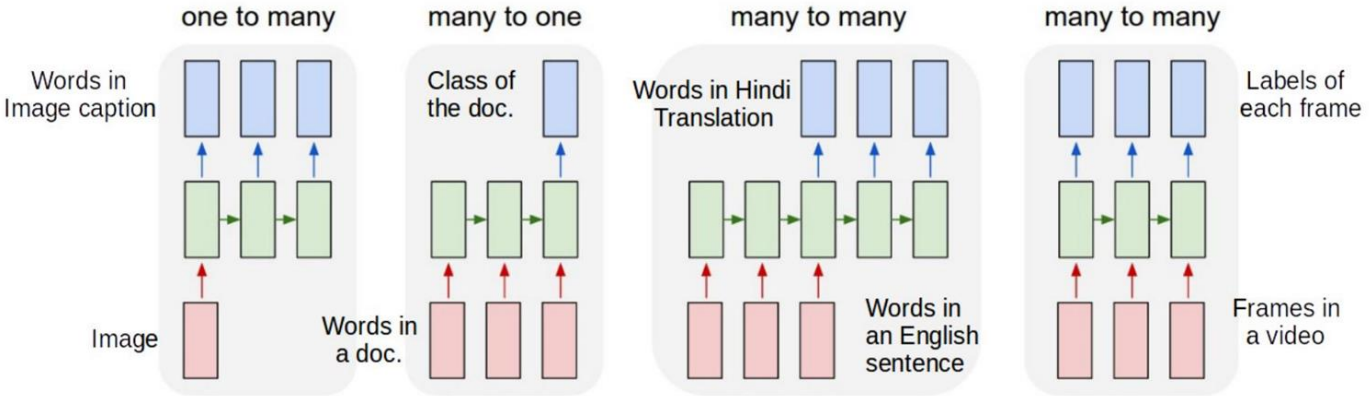


- Experiments on machine translation, abstractive summarization and language modeling suggest that this method can yield comparative results with other self-attention and convolution based competitive methods.

Introduction

- Sequence modeling is a fundamental task in ML
- It's the process of learning how to combine timesteps to form representations of higher abstraction.

[Karpathy, 2015]

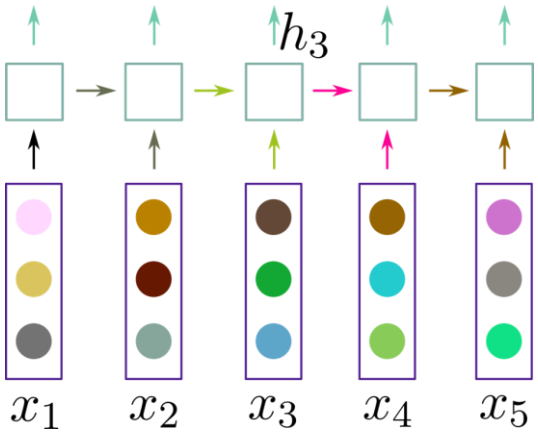


- Many applications such as machine translation, POS tagging, sentiment classification, video processing, time-series etc.

Sequence Modeling Approaches

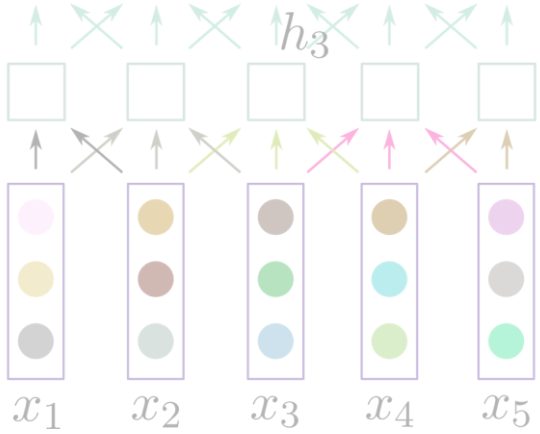
RNNs Hochreiter and Schmidhuber, 1997
Cho et al., 2014

$$h_t = f(x_t, h_{t-1})$$



CNNs Gehring et al, 2017
Wu et al., 2019

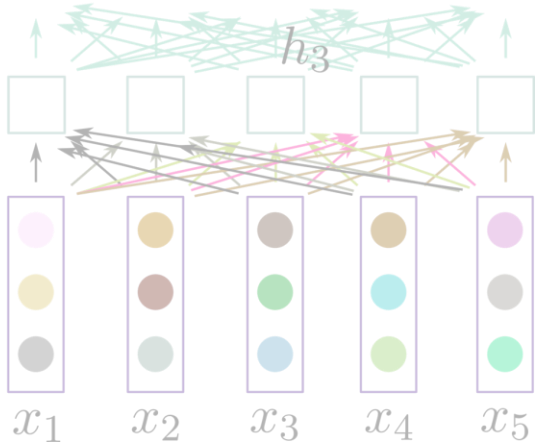
$$h_t = f(x_{t-\lfloor k/2 \rfloor}, \ldots, x_{t-\lfloor k/2 \rfloor+k})$$



Self-Attention Cheng et al, 2016
Vaswani et al., 2017

$$h_t = \sum_{i=1}^{n} a_{t,i} \cdot x_i$$

$$a_{t,i} = \frac{\exp(f(x_t, x_i))}{\sum_{j=1}^{n} \exp(f(x_t, x_j))}$$



Sequence Modeling Approaches

RNNs Hochreiter and Schmidhuber, 1997
Cho et al., 2014

CNNs Gehring et al, 2017
Wu et al., 2019

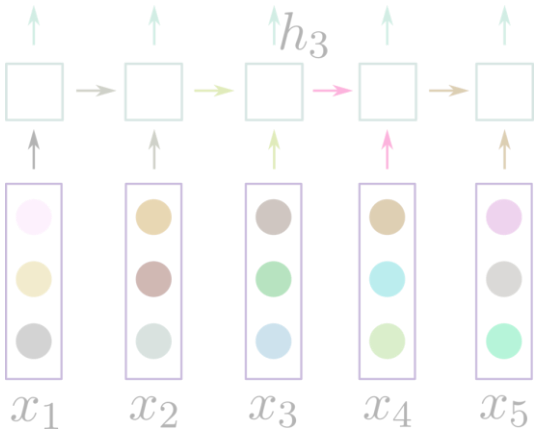
Self-Attention Cheng et al, 2016
Vaswani et al., 2017

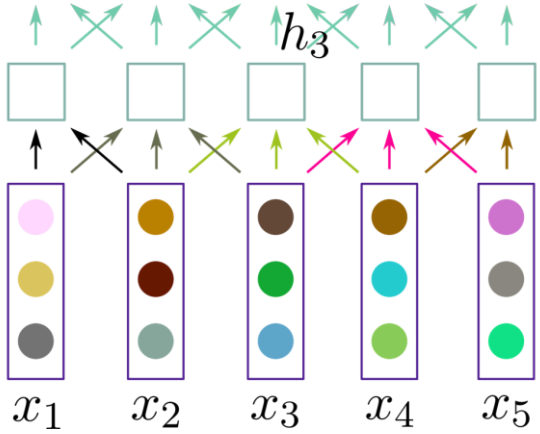
$$h_t = f(x_t, h_{t-1})$$

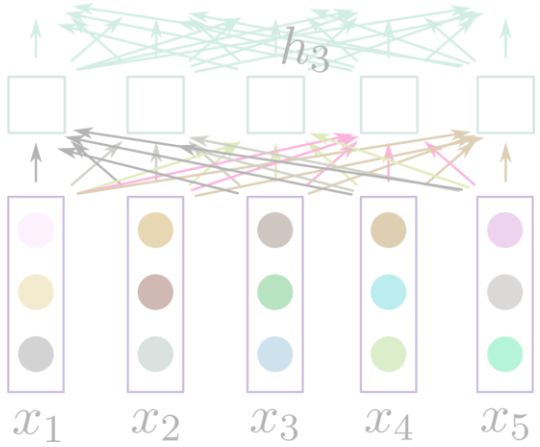
$$h_t = f(x_{t-\lfloor k/2 \rfloor}, \ldots, x_{t-\lfloor k/2 \rfloor + k})$$

$$h_t = \sum_{i=1}^{n} a_{t,i} \cdot x_i$$

$$a_{t,i} = \frac{\exp(f(x_t, x_i))}{\sum_{j=1}^{n} \exp(f(x_t, x_j))}$$



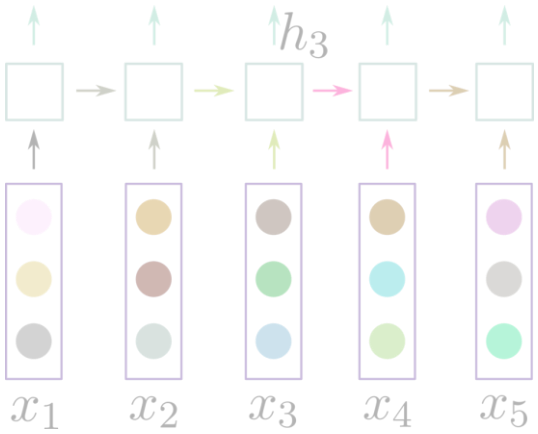




Sequence Modeling Approaches

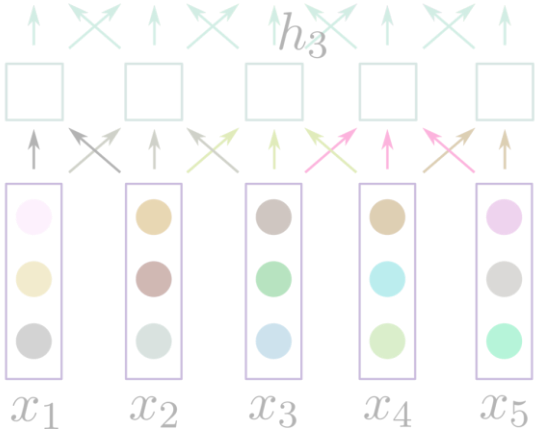
RNNs Hochreiter and Schmidhuber, 1997
Cho et al., 2014

$$h_t = f(x_t, h_{t-1})$$



CNNs Gehring et al, 2017
Wu et al., 2019

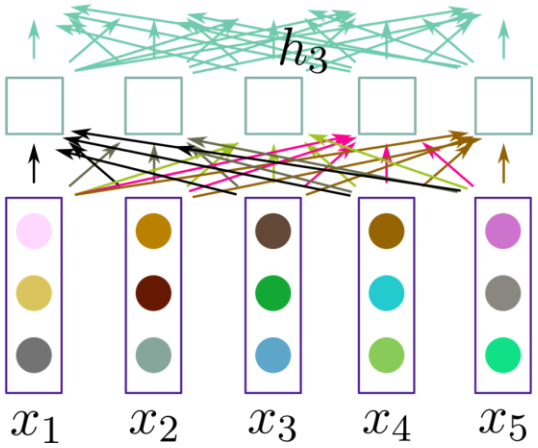
$$h_t = f(x_{t-\lfloor k/2 \rfloor}, \ldots, x_{t-\lfloor k/2 \rfloor+k})$$



Self-Attention Cheng et al, 2016
Vaswani et al., 2017

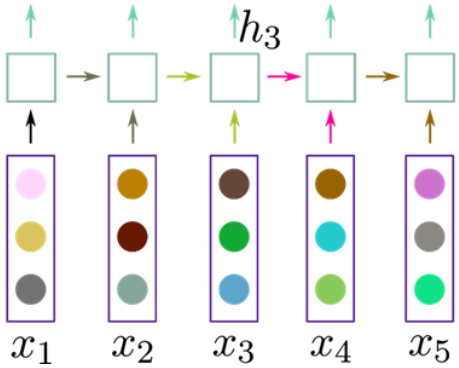
$$h_t = \sum_{i=1}^{n} a_{t,i} \cdot x_i$$

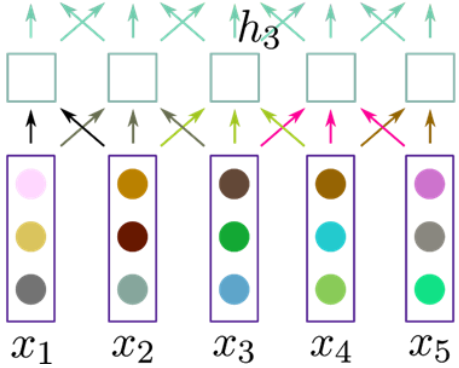
$$a_{t,i} = \frac{\exp(f(x_t, x_i))}{\sum_{j=1}^{n} \exp(f(x_t, x_j))}$$

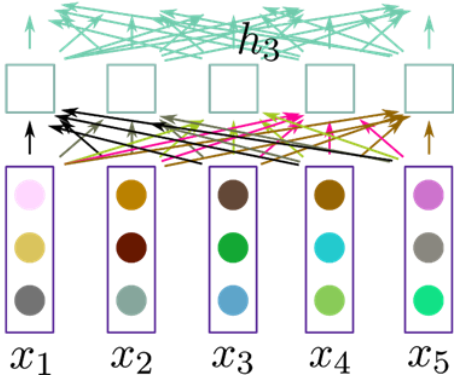


Comparison

	RNNs	CNNs	Self-Attention
Parallel	✗	✓	✓
Infinite Context	✓	✗	✓
Time Complexity	$O(n)$	$O(k \cdot n)$	$O(n^2)$







Motivation

Currently, **self-attention** is considered **vital** for modern sequence learning approaches.

- Self-attention is **expensive**. It has quadratic time complexity.
  - Hard to be deployed on devices with limited hardware (i.e. edge devices)
- Dynamic Convolutions [Wu et al. 2019] showed that you can achieve good results using a **limited context window**.
  - Still relies on a special type of attention (i.e. dynamic value-based attention)

Research Questions

- Q1: Is (self-)attention critical to get good performance?
- Q2: Can we reduce the time-complexity to $O(n)$ using a parallelizable non-autoregressive method?

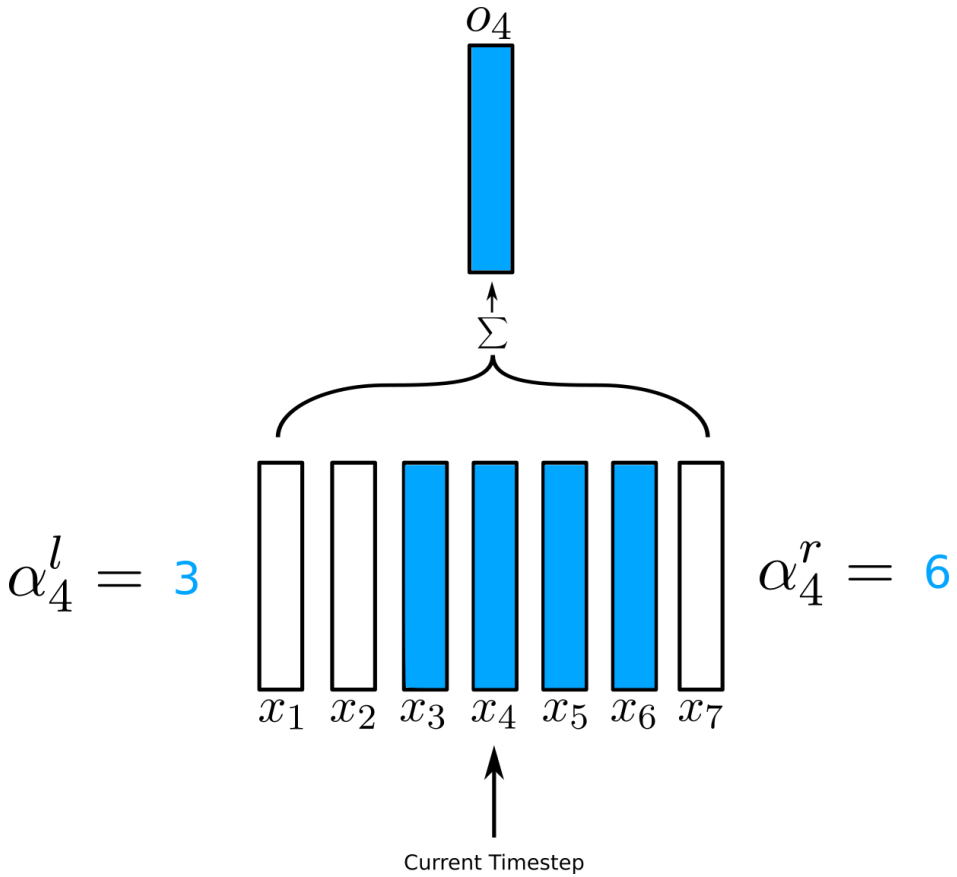
One-dimensional Large Kernel Convolution

- One of the simplest ways to model a sequence of representations is to aggregate the appropriate number of vector representations together.

$$o_i = \sum_{j=\alpha_i^l}^{\alpha_i^r} x_j,$$

where $1 \leq \alpha_i^l \leq i \leq \alpha_i^r \leq n$ are the left and right offsets (boundaries).

One-dimensional Large Kernel Convolution



Summed-area Table

- Applying the previous aggregation can be slow because we compute the same aggregations again and again.
- To address this issue we can use the summed-area table (integral image operation).
- Let $\mathcal{S} = \{\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n\}$ be the summed-area table computed using

$$\begin{cases} \mathcal{S}_0 = 0, \\ \mathcal{S}_i = \mathcal{S}_{i-1} + x_i, \quad 1 \leq i \leq n. \end{cases}$$

- The above operation can be efficiently parallelized with complexity $O(\log n)$ using the parallel prefix sum algorithm.
- Given the left (α_i^l) and right (α_i^r) offsets, we can compute O_i using the summed-area table in $O(1)$ time:

$$o_i = \mathcal{S}_{a_i^r} - \mathcal{S}_{a_i^l - 1}$$

Time-aware Large Kernel Generation

- So far, we assumed that α_i^l and α_i^r are given.
- Ideally, we want to learn to generate these offsets for each input timestep.
- We can't directly predict the index which corresponds to the offset word:
  - Indexes are positive unbounded integers; $i \in [1, N]$
- We address this issue using relative offsets.
- We generate these relative offsets using

$$\tilde{a}_i^{\{l,r\}} = \sigma(f^{\{l,r\}}(x_i)) \in [0, 1]$$

where $f^{\{l,r\}} : \mathbb{R}^d \rightarrow \mathbb{R}$

Offsets Interpolation

- Convert the relative offsets to absolute by using

$$a_i^l = i - \tilde{a}_i^l \cdot l_{\max}$$
$$a_i^r = i + \tilde{a}_i^r \cdot r_{\max}$$

where $\{l_{\max}, r_{\max}\} \in \mathbb{Z}_{\geq 0}$ are the maximum allowed tokens to the left and to the right.

- We can't directly use the absolute indexes because they are real values.
- We use linear interpolation to approximately generate $\mathcal{S}_{a_i^l - 1}$ and $\mathcal{S}_{a_i^r}$ directly:

$$\mathcal{S}_{a_i^l - 1} = \gamma^l \cdot \mathcal{S}_{a_i^l - 1} + (1 - \gamma^l) \cdot \mathcal{S}_{a_i^l - 1}$$
$$\mathcal{S}_{a_i^r} = (1 - \gamma^r) \cdot \mathcal{S}_{a_i^r} + \gamma^r \cdot \mathcal{S}_{a_i^r}$$

Output Normalization

- The proposed method works well when used with shallow models.
- Aggregating many representations together can lead to disproportional magnitude on the representation values passed to the next layers.
- Solution: Normalize by the maximum window length

$$\tilde{o}_i = o_i \cdot \left(\frac{1}{l_{\max} + r_{\max} + 1} \right)$$

- To further increase the performance, we apply dropout to the generated relative offsets $\tilde{a}_i^{\{l,r\}}$
  - Set relative offset to zero which effectively cancels the expansion of the window towards that direction.
  - Forcing the model to produce smaller windows to robustify the importance of the number of tokens that are needed to model a timestep.

Multi-headed Kernels

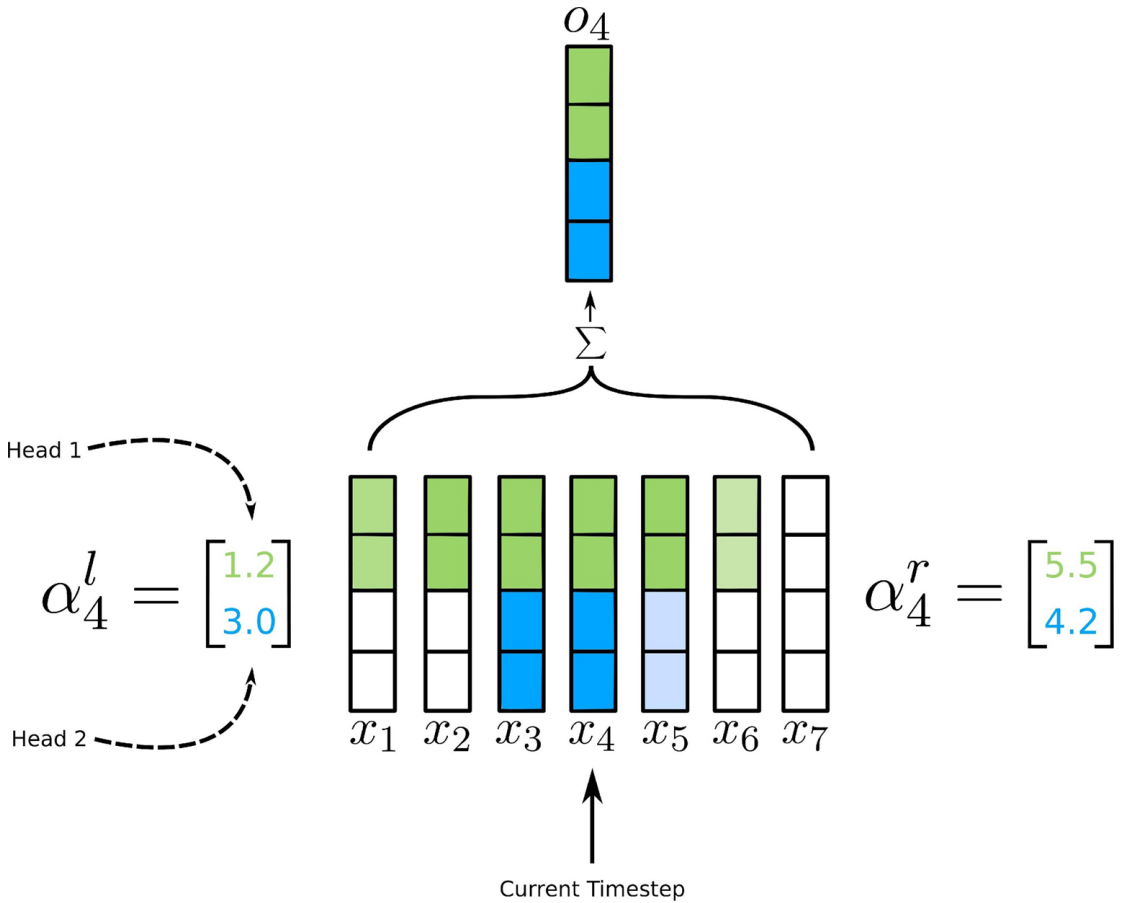
- Similar to MHSA, we introduce multiple heads.
- We tie every subsequent number of $R = \frac{d}{H}$ channels together and group the channels into H groups.

$$\tilde{\alpha}_i^{\{l,r\}} = (f^{\{l,r\}}(\hat{x}_i)) \in [0,1]^H$$

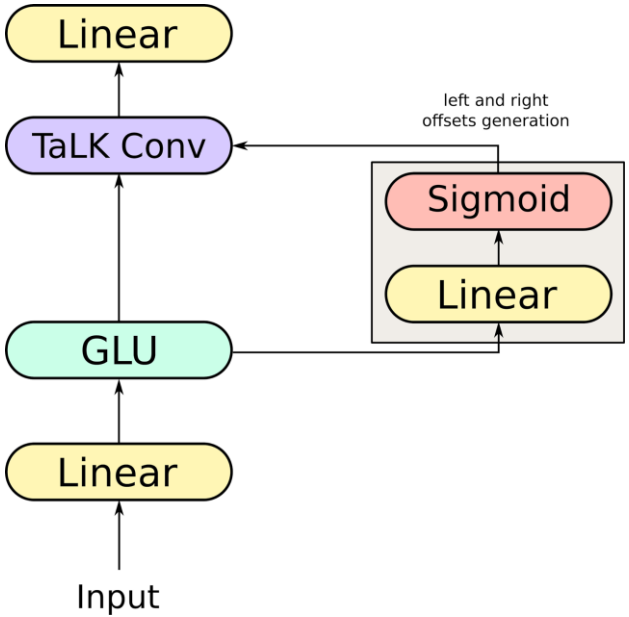
where $f^{\{l,r\}} : \mathbb{R}^{H \times R} \rightarrow \mathbb{R}^H$.

- This helps to further increase the expressivity and diversity of the representation of each timestep.

The TaLK Convolution Operation



Architecture & Implementation



- We implemented our own CUDA primitives to support the TaLK Convolution operation.

Computational Complexity

Table 4.1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension and k is the kernel size of convolutions.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Recurrent [8]	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional [11, 21]	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$ or $O(n/k)$
Self-Attention [13]	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Dynamic Convolutions [12]	$O(k \cdot n \cdot d)$	$O(1)$	$O(n/k)$
TaLK Convolutions (Ours)	$O(n \cdot d)$	$O(log(n))$	$O(n/(l_{\max} + r_{\max} + 1))$

Machine Translation

Table 2. Machine translation accuracy in terms of BLEU for WMT En-De and WMT En-Fr on newstest2014.

Model	Param (En-De)	WMT En-De	WMT En-Fr
Gehring et al. (2017)	216M	25.2	40.5
Vaswani et al. (2017)	213M	28.4	41.0
Ahmed et al. (2017)	213M	28.9	41.4
Chen et al. (2018)	379M	28.5	41.0
Shaw et al. (2018)	-	29.2	41.5
Ott et al. (2018)	210M	29.3	**43.2**
Wu et al. (2019)	213M	**29.7**	**43.2**
TaLK Convolution (Ours)	209M	29.6	**43.2**

Table 3. Machine translation accuracy in terms of BLEU on IWSLT De-En.

Model	Param	IWSLT De-En
Deng et al. (2018)	-	33.1
Vaswani et al. (2017)	47M	34.4
Wu et al. (2019)	43M	35.2
TaLK Convolution (Ours)	42M	**35.5**

Abstractive Summarization & Language Modeling

Table 4. Results on CNN-DailyMail abstractive summarization.

Model	Param	Rouge-1	Rouge-2	Rouge-L
LSTM (Paulus et al., 2018)	-	38.30	14.81	35.49
CNN (Fan et al., 2018)	-	39.06	15.38	35.77
Self-Attention Baseline (Wu et al., 2019)	90M	39.26	15.98	36.35
Lightweight Convolution (Wu et al., 2019)	86M	39.52	15.97	36.51
Dynamic Convolution (Wu et al., 2019)	87M	39.84	16.25	36.73
TaLK Convolution (Standard)	59M	40.03	18.45	36.13
TaLK Convolution (Deep)	83M	**40.59**	**18.97**	**36.81**

Table 5. Test perplexity on WikiText-103.

	Param	Test
Grave et al. (2017)	-	40.8
Dauphin et al. (2017)	229M	37.2
Merity et al. (2018)	151M	33.0
Rae et al. (2018)	-	29.2
Baevski & Auli (2019)	247M	**20.5**
Dynamic Convolution	255M	25.0
TaLK Convolution (Ours)	240M	23.3

Model Ablation

Table 7. Ablation on IWSLT De-En validation set. (+) indicates that a result includes all preceding features.

Model	Param	BLEU
TaLK Convolution (a_i^l, a_i^r=1x7, H=1)	42M	diverges
+ Output Normalization	42M	35.70 $\pm$ 0.1
+ Increasing Max Offsets (a_i^l, a_i^r=1,3,7,15x4)	42M	36.23 $\pm$ 0.1
+ Offsets Dropout (p=0.1)	42M	36.37 $\pm$ 0.05
+ Fully-headed Kernels (H=512)	47M	36.51 $\pm$ 0.07
+ Multi-headed Kernels (H=4)	42M	**36.65** $\pm$ 0.05
+ Replacing Swish with ReLU	42M	36.21 $\pm$ 0.05

Encoding Inference Speed Comparison

Table 5.11: Throughput and memory consumption decrease measured for different sequence lengths (n) on a batch of size 10 with each token being represented with $d = 1024$ and $H = 16$. Throughput is calculated across 100K iterations of a single input encoding execution for each method. Memory decrease is computed as how many times less memory we need to encoding the input embedding compared to Self-Attention. Larger numbers indicate better performance.

Method	$n = 10$		$n = 100$		$n = 1,000$		$n = 10,000$	
	iter/sec	Mem. ↓	iter/sec	Mem. ↓	iter/sec	Mem. ↓	iter/sec	Mem. ↓
Self-Attention	4576	1x	3437	1x	102	1x	OOM	1x
DynamicConv ($k = 3$)	3739	1x	3308	0.99x	443	2.8x	45	25.4x
DynamicConv ($k = 31$)	4535	0.97x	3860	1x	325	2.7x	29	20.2x
TaLK Convolution	**9686**	**1.1x**	**6126**	**1.1x**	**898**	**3.1x**	**92**	**26.4x**

Conclusion

- We introduced a new way of doing sequence modeling that has $O(n)$ time complexity.
- The results show that the proposed method can perform on par with transformers and dynamic convolutions without using self-attention or a variant of it.
- In the future, we will do more research on how to apply TaLK Convolutions in a non-contiguous way.

Thank you!

github.com/lioutasb/TaLKConvolutions